Preventing Keystroke Based Identification in Open Data Sets

Leinonen, Juho

ACM
2017-04-12

Leinonen , J , Ihantola , P & Hellas , A 2017 , Preventing Keystroke Based Identification in Open Data Sets . in Proceedings of the Fourth (2017) ACM Conference on Learning @ Scale . ACM , New York, NY , pp. 101-109 , ACM Conference on Learning at Scale , Cambridge, MA , United States , 20/04/2017 . https://doi.org/10.1145/3051457.3051458

http://hdl.handle.net/10138/316009
https://doi.org/10.1145/3051457.3051458

acceptedVersion

Preventing Keystroke Based Identification in Open Data Sets

Juho Leinonen
University of Helsinki
Department of Computer Science
Helsinki, Finland
juho.leinonen@helsinki.fi

Petri Ihantola
Tampere University of Technology
Department of Pervasive Computing
Tampere, Finland
petri.ihantola@tut.fi

Arto Hellas
University of Helsinki
Department of Computer Science
Helsinki, Finland
arto.hellas@cs.helsinki.fi

ABSTRACT

Large-scale courses such as Massive Online Open Courses (MOOCs) can be a great data source for researchers. Ideally, the data gathered on such courses should be openly available to all researchers. Studies could be easily replicated and novel studies on existing data could be conducted. However, very fine-grained data such as source code snapshots can contain hidden identifiers. For example, distinct typing patterns that identify individuals can be extracted from such data. Hence, simply removing explicit identifiers such as names and student numbers is not sufficient to protect the privacy of the users who have supplied the data. At the same time, removing all keystroke information would decrease the value of the shared data significantly.

In this work, we study how keystroke data from a programming context could be modified to prevent keystroke latency based identification whilst still retaining information that can be used to e.g. infer programming experience. We investigate the degree of anonymization required to render identification of students based on their typing patterns unreliable. Then, we study whether the modified keystroke data can still be used to infer the programming experience of the students as a case study of whether the anonymized typing patterns have retained at least some informative value.

We show that it is possible to modify data so that keystroke latency based identification is no longer accurate, but the programming experience of the students can still be inferred, i.e. the data still has value to researchers. In a broader context, our results indicate that information and anonymity are not necessarily mutually exclusive.

L@S 2017, April 20 - 21, 2017, Cambridge, MA, USA

ISBN 978-1-4503-4450-0/17/04…$15.00

DOI: http://dx.doi.org/10.1145/3051457.3051458

CCS Concepts

•**Security and privacy** → **Pseudonymity, anonymity and untraceability; Data anonymization and sanitization; Privacy protections;** •**Information systems** → *Data mining;* •**Social and professional topics** → Computing education;

Author Keywords

data privacy; data anonymization; keystroke dynamics; programming experience inference; source code snapshots

INTRODUCTION

Nowadays, a lot of data is shared openly for replication studies and novel analysis on existing data [3, 6, 18]. Still, privacy issues often prevent companies, governments, and (educational) institutions from sharing the data that they have collected [10]. Sharing non-anonymized data that could be used to identify individuals would violate the privacy of the users or parties from which the data has been collected. Anonymizing data by simply removing parts of the data – attributes – may not be sufficient as latent factors that can be used to identify individuals may exist.

Attributes that are not identifiers by themselves, but can be used for identification together with other attributes are called quasi-identifiers [10]. For example, Daries et al. [5] studied anonymization of MOOC data from a social science perspective, and defined the country, gender, age and level of education of a participant as quasi-identifiers. Similarly, keystroke timings found in programming snapshots are quasi-identifiers: a single keystroke timing does not reveal the identity of the typist, but together the timings can be used to construct a typing profile that can be used for identification [7, 11, 15, 21, 24]. Longi et al. [21] have showed that individual programmers can be identified from source code snapshots based on the times that the programmers take to move from one key to another, i.e. the typing pattern.

From a computer science education viewpoint, having fine-grained keystroke data provides a detailed picture of the students' learning process [30]. Research carried by Vihavainen et al. [31] found that keystroke level data can be used to conduct studies that are not possible with more coarse-grained data. Such data can also be used for inferring the programming experience of students [20].

However, it is rare to include keystroke data in open data sets. While source code snapshot data is publicly available by, for example, the Blackbox-project [3], the data does not include keystroke level data. Thus, keystroke timing based studies (e.g. [2, 7, 20, 29]) are presently hard to replicate because such data is rarely collected and available. This has been acknowledged as a problem and there seems to be pressure (and a trend) for publishing more fine-grained learning data than what is available today [14]. Al-Zubidy et al. note that replication studies are essential for theory building and are therefore concerned about the lack of replication studies in the computer science education field [1].

Daries et al. [5] showed that in a social science context, the value of data can degrade significantly in the anonymization process – results on anonymized data differ from results on non-anonymized data. In this work, we study whether there is a similar effect in anonymizing source code snapshot data. More specifically, we investigate whether keystroke timing data in source code snapshots can be modified in a way that prevents typing pattern based identification, whilst other valuable information can still be inferred from the anonymized keystroke timing data. While identification could also be possible from other identifiers found in keystroke data such as text content (variable names, class names, etc.), we focus on preventing identification based on keystroke timings. Being able to modify keystroke timings so that they cannot be used for identification would remove a quasi-identifier from the data, which would maintain the possibility that anonymized keystroke timings could be included in open data sets and used for research.

It has been previously shown that programming experience can be inferred from keystroke timings to a degree [20]. Thus, we conduct a case study where programming experience is the valuable information we wish to be able to infer from anonymized keystroke timing data. Inferring programming experience from keystroke timings can be useful on data sets that do not include programming background information or with students who have not answered a background survey [20]. We conduct experiments using two anonymization procedures and compare identification accuracies with different degrees of anonymization. Furthermore, we seek to find a balance where programmers could not be identified based on keystroke timings but programming experience could still be inferred. Being able to infer programming experience but not the individuals would suggest that there is value for researchers in the data, while the privacy of the individuals would be preserved. This is a step towards releasing fine-grained source code snapshot data openly to others.

This article is organized as follows. First, we provide a summary of previous work related to identification using keystroke dynamics, inferring valuable information from keystroke timings, and data anonymization. Next, we outline our research methodology and data. Then, our experiments and their results are presented. Finally, a discussion of the results and conclusions are presented.

RELATED WORK

Here, we visit three streams of related work. First, we discuss articles where keystroke data has been used to infer the identity of a user, then we discuss articles related to inferring other information in addition to identity from keystroke timings, and finally, we visit data anonymization from a data sharing perspective.

Keystrokes and identity

Information recorded from typing, such as the duration of keystrokes, pressure of keystrokes, and keystroke latencies, has been used for identification purposes [7, 11, 15, 21, 22, 24]. From these especially the keystroke latencies between pairs of keys – digraphs – have been used extensively [7,11,21,24]. For example, a study by Longi et al. [21] shows that the identity of programmers can be detected from keystroke data recorded during programming sessions. Using data from two separate courses, they observed that linking the students from one course to another – when using full course data from both courses – could be done with 98.6% accuracy. They note that keystroke identification is an especially convenient way of authentication in Massive Online Open Courses (MOOCs) as it is irrespective of location and thus perfect for distance learning. The MOOC platform Coursera is already using keystroke identification as they collect typing samples from students seeking to acquire a verified certificate for completing a course [4].

Identification results often vary significantly based on the data used. For example, in a study by Monrose and Rubin with 46 participants, the identification accuracy decreased significantly from 79% with transcribed text to 21% with free text [24]. It was suggested that this could be explained by the writer having to think about what they were going to write instead of just writing whatever was given to them. However, Killourhy and Maxion found no significant difference in classification results when using transcribed or free text [17].

Keystroke analysis has been applied successfully for identifying students in online exams [19, 22, 27]. Using data from 30 students taking examinations in a business school, Monaco et al. were able to correctly identify all the students [22]. Likewise, Leinonen et al. [19] were able to identify a large portion of the students in programming exams where students code on a computer. They showed that students can be identified quite reliably in both controlled and uncontrolled exam environments. In the controlled exam, the students were in a computer lab at the university and in the uncontrolled exam they could be in whatever setting they found most comfortable, e.g. at home.

Inferring information based on keystroke timing

In addition to identification and authentication, keystroke timings can be used for inferring other information. Thomas et al. have studied the relationship between keystroke latencies and programming performance [29]. They categorized digraphs, i.e. character pairs, into seven categories based on their type and calculated the mean latency by category. They found statistically significant correlations between the mean latencies of some categories and exam results. An explanation they

provide is that skilled programmers type some digraphs faster than novice programmers.

More recently, Leinonen et al. [20] partially replicated the study by Thomas et al. [29] by analyzing the relationship between digraph latencies and programming performance. Furthermore, they described an experiment where they sought to identify students' past programming experience from keystroke latencies. Leinonen et al. note that inferring programming experience from keystroke latency data can be more reliable than a background questionnaire as some students may choose not to answer such questionnaires. After performing feature selection on digraph latencies and experimenting with a set of classifiers, they observed up to 77% classification accuracy and a Matthew's Correlation Coefficient of 0.54 in predicting whether a student had programmed previously or not. As an example, they showed that on average, experienced programmers move faster from the key *i* to the key +, i.e. experienced programmers type the digraph *i*+, faster than novice programmers and thus at least partially confirmed the suggestion by Thomas et al. Intuitively, this makes sense as the digraph *i*+ is something programmers type often when incrementing an index variable, while it rarely occurs in regular text.

Additionally, keystroke analysis has been used to detect boredom and engagement [2], stress [33], and emotional states [9].

Data anonymization

Anonymity in data is often achieved by removing attributes from the data [10, 25, 28], reducing the accuracy of the data, e.g. by grouping and smoothing [13, 16] and by adding noise or fake information [8, 16]. Sun and Upadhyaya have developed a rule-based data sanitization method to remove sensitive information such as social security numbers from keystroke data [28].

Fung et al. outline four different types of attributes in data which reserve privacy: explicit identifiers, quasi-identifiers, sensitive attributes, and non-sensitive attributes [10]. Quasi-identifiers are attributes that are not identifiers by themselves, but can be used for identification together with other quasi-identifiers. As an example of anonymizing data by removing explicit identifiers and quasi-identifiers, network measurement data could be anonymized by removing attributes such as packet payloads and ip-addresses [25]. Daries et al. [5] analyzed the anonymization of data collected on MOOCs. They found two explicit identifiers – username and ip-address – and six quasi-identifiers – country, age, gender, and level of education of a participant as well as course id and the amount of forum posts – in their data and removed them. Similarly, explicit identifiers such as student numbers and quasi-identifiers such as keystroke timestamp information could be removed from source code snapshot data. However, removing quasi-identifiers also reduces the value of the data as information that is possibly relevant for research can be lost in the process as Daries et al. noticed [5]. Thus, modifying such data in a way that preserves privacy but yields possibility for research would be optimal.

In addition to removing attributes, other approaches for preserving anonymity have been suggested. For example, He et al. [13] suggested anonymization of set-valued data by distributing the data into buckets. Their work was motivated by the fact that the previously suggested approaches work well only if a subject is associated with a single sensitive value at a time, which does not suit set-valued data well. Similarly, Samarati et al. suggested replacing values in the data by semantically consistent less precise alternatives [26], i.e. generalization or rounding. A challenge here is to find an optimal degree of anonymization where data is minimally distorted while identification of subjects is still made improbable.

Recently, Monaco and Tappert developed two obfuscation strategies in the context of a third party continuously recording keystroke data [23]. They were able to decrease identification accuracy on average by 20% by adding a 25 ms random delay to the keystroke events and found that a delay of 500 ms was needed to reduce identification accuracy by half. In the context of a constant flow of keystrokes, there is a constraint that the anonymization should not affect the user experience, e.g. an added delay can not be noticeably long. However, in our context of open data sets there is no such constraint, which allows calculating optimal degrees of anonymization post hoc.

METHODOLOGY

In this section, we outline our research questions, the context of the data we use, and our research methodology.

Research questions

In this work, we seek to determine how different degrees of anonymization of programming course data affects attributes that can be inferred from typing profiles. Our research questions are:

RQ 1. How does anonymization by rounding keystroke average latencies affect identification accuracy?

RQ 2. How does anonymization by bucketing affect identification accuracy?

RQ 3. How does anonymization affect inferring programming experience from typing profiles?

With the first research question, we seek to determine how rounding average latencies can be used to anonymize keystroke data. With the second research question, we explore whether splitting the data into even-sized buckets works for anonymization. Finally, with the third research question, we examine the extent of anonymization one can perform whilst still retaining information about programming experience. We are interested in finding an optimal amount of anonymization where identification is no longer practical, but programming experience can still be inferred.

Context

The data used in the experiments come from two similar introductory Java programming courses held in the autumns of 2014 and 2015 at University of Helsinki. One of the authors of this work was responsible for organizing the courses. Both courses lasted for 7 weeks. The courses taught the students

programming basics such as variables, loops, input, and output. Both data sets were used in the identification experiments, but only the autumn 2014 course had information available on students' programming background, and therefore was the only one included in the programming experience experiments. 41.2% of the students had at least some programming experience and 58.8% had none.

The students used an integrated development environment (IDE) for working on the course assignments. The IDE used the Test My Code -plugin [32] which records a snapshot for each action where the student modifies the source code. The snapshots have a nanosecond level timestamp in addition to keystroke information. Students could turn the data gathering mechanism in the environment off if they chose to – data for this study was provided on a voluntary basis and no incentives were given to students who provided the data.

Preprocessing

For preprocessing the keystroke data, we followed the procedure outlined in the study by Longi et al. [21]. Only digraphs with latencies between 10 ms and 750 ms were included as first done by Dowland and Furnell [7]. The lower bound is necessary to eliminate auto-completion events from the IDE. The upper bound is needed to only capture the subconscious typing rhythm of the student and to remove any breaks they might take while working on an exercise.

Since the typing profiles are built with average latencies, we required that a student should have at least 5 occurrences of any digraph used to build their typing profile. If the student had only typed a digraph under 5 times, the average latency for that digraph was excluded from the student's typing profile. Snapshots where multiple characters were added at the same time were discarded as they were almost exclusively copy-paste events.

After preprocessing, there were 199 students left in the autumn 2014 data set and 153 in the autumn 2015 data set.

Identification

For the identification experiments, we use the acceptance threshold method introduced by Longi et al. [21] where a match in the top k closest training set samples is considered correct for a specific test sample. The idea behind this is that exact identification is not always mandatory. For example, for authentication in online exams, it is sufficient to be quite sure that the students are who they claim to be.

To build the typing profiles, the average latency between two specific characters was calculated for all character pairs, i.e. digraphs, for each student in the data. If a student had not typed a digraph, the missing value was replaced with the student's average typing speed.

For both data sets used in the identification experiments, we chose to build the typing profiles in the training set from the first six weeks of exercises and used the data from exercises of the last week as the test set. To determine if a test sample was correctly identified, we calculated the euclidean distance to each training set sample. We then sorted the training set samples based on the distance from the test sample. We used an acceptance threshold of $k = 10$, and thus regarded the student to be correctly identified if their typing profile was in the top 10 closest training set matches.

Programming experience inference

Earlier research indicates that the Bayesian Network and Random Forest classifiers have good performance at classifying students in the context of inferring programming experience from typing profiles [20]. Therefore, we classify the students into two groups: those with some programming experience and those with none using the Bayesian Network, Random Forest, and ZeroR classifiers. The ZeroR classifier is a majority class classifier which will classify every sample to the majority class, and is therefore good as a baseline against which the performance of the other two classifiers can be measured. The classification accuracy is evaluated using 10-fold cross-validation.

Anonymization by rounding

We use an anonymization technique similar to generalization [26] where the values in the data are rounded to reduce identification accuracy (RQ1). To investigate how rounding the average latencies in typing profiles affects programmer identification and classification based on programming experience, we modified the latencies using Equation 1. It rounds the latency z to the nearest x, where x is the number of milliseconds given to the anonymization function as a parameter. The resulting value y is then used instead of the original value z in the construction of the typing profile. The aim is to reduce the accuracy of the data, hopefully reducing identification accuracy in the process, which would anonymize the data. We studied how identification accuracy deteriorates when the value of x is increased.

$$y = x * round(z/x) \tag{1}$$

Equation 1 essentially distributes the average latencies into buckets. For example, if x is 100 milliseconds, all latencies will be rounded to the nearest multiple of 100. This leads to all latencies between 0 and 50 ms being rounded to 0 and distributed to the first bucket, all latencies between 50 and 150 ms being rounded to 100 and distributed to the second bucket, and so on.

After rounding the average latencies, the data was normalized to reduce the effect of digraphs with large average latencies on the distance calculations.

Anonymization by distributing the data into even-sized buckets

The buckets that result from the rounding method are not equal in size: the size of the first bucket is half the size of the subsequent buckets. Motivated by this we analyzed whether distributing data into even-sized buckets could be used for anonymizing keystroke data (RQ2). We modified the average latencies in the data by first increasing each latency z by half of the size b of the buckets using Equation 2, and then rounding each latency z_1 to the nearest x, where x is the current bucket size b using Equation 3. The resulting value y is then used

instead of the original value z in the construction of the typing profile.

$$z_1 = z + (b/2) \tag{2}$$

$$y = b * round(z_1/b) \tag{3}$$

The only difference between this method and the rounding method is that this method distributes the data into even-sized buckets. For example, if we have buckets of 100 milliseconds, we want all latencies between 0 and 100 milliseconds to be in the same bucket. Now, any latency between 0 and 100 ms will first be incremented by 50 ms (half the bucket size), leading to a distribution between 50 and 150 ms. Then, the latencies will be rounded to the nearest multiple of 100 milliseconds (the bucket size), which in the case of values between 50 and 150 milliseconds is 100 milliseconds. The procedure is then repeated for all values between 100 and 200 milliseconds, etc.

Again, after rounding the average latencies, the data was normalized to reduce the effect of digraphs with large average latencies on the distance calculations.

Feature selection

For exploring how identification accuracy suffers when the data is anonymized (RQ1 & RQ2), the 25 most common digraphs were used to construct the typing profiles as first done by Leinonen et al. [19]

In the feature selection for the programming experience data set (RQ3), we followed the procedure outlined by Leinonen et al. [20] for inferring programming experience from typing profiles. Features with no data (e.g. digraphs that no student had typed) were removed. Then, the WEKA Data Mining toolkit [12] was used for feature selection. Out of more than 10000 initial features, less than 50 features were left in each data set after the feature selection.

EXPERIMENTS AND RESULTS

In this section, we describe the experiments we conducted to answer each of the research questions and the results of the experiments.

Identification experiments

To answer the first two research questions *"How does anonymization by rounding keystroke average latencies affect identification accuracy?"* and *"How does anonymization by bucketing affect identification accuracy?"*, we calculated identification accuracies with different degrees of anonymization.

The results of the experiments are presented in Table 1. The millisecond values in the first column represent the rounding for RQ1 and the bucket size for RQ2. The 0 ms row shows the identification accuracy without modifications (rounding or bucketing), i.e. without anonymization.

When using rounding for anonymization, identification accuracies in both data sets deteriorate in the first two 100 ms steps, but then improve or stay equal in the next two steps. After that they start declining again. The unexpected value of 6.5% in the rounding experiment of the 2014 data set when rounding to 200 ms is studied in further detail later.

Table 1. Identification accuracy percentages with different rounding precisions and bucket sizes between 0 ms (no anonymization) and 600 ms.

Method	**Rounding**		**Buckets**	
Data	2014	2015	2014	2015
0 ms	98.0	97.8	98.0	97.8
100 ms	81.7	81.3	26.1	31.7
200 ms	6.5	56.8	12.4	15.8
300 ms	72.5	67.6	6.5	7.2
400 ms	77.1	67.6	6.5	7.2
500 ms	6.5	7.2	6.5	7.2
600 ms	6.5	7.2	6.5	7.2

Table 2. Programming experience classification accuracy percentages with different rounding precisions and bucket sizes.

Method	**Rounding**		
Classifier	Bayes Net	Random Forest	ZeroR
0 ms	75.4	73.9	58.8
100 ms	73.9	75.4	58.8
200 ms	73.9	72.4	58.8
300 ms	73.9	70.9	58.8
400 ms	68.3	73.4	58.8
500 ms	73.9	73.9	58.8
600 ms	70.4	71.4	58.8
Method	**Buckets**		
Classifier	Bayes Net	Random Forest	ZeroR
0 ms	75.4	73.9	58.8
100 ms	73.4	73.4	58.8
200 ms	71.4	75.4	58.8
300 ms	70.4	71.4	58.8
400 ms	61.3	69.8	58.8
500 ms	64.3	67.3	58.8
600 ms	58.8	60.3	58.8

Using buckets for anonymization, identification accuracies in both data sets deteriorate with every 100 ms step and reach their lowest values already after three steps. These results are different from the results of the rounding anonymization method, where the lowest values were only attained after 5 steps and at the third step mark the identification accuracies were still quite high at around 70% accuracy compared to around 7% accuracy with the bucket approach.

Inferring programming experience from anonymized data

To answer the third research question, *"How does anonymization affect inferring programming experience from typing profiles?"*, we measured classification accuracies with different amounts of anonymization using both the rounding method and the bucket method and multiple classifiers.

Table 2 shows the classification accuracy results with different amounts of anonymization. With the rounding method, classification accuracies deteriorate slightly with each step, although there are exceptions. We do not observe a similar effect as

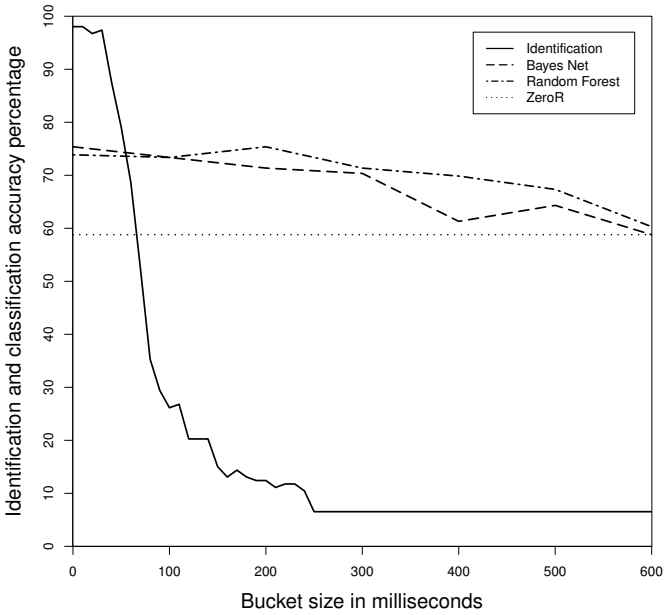

Figure 1. Identification (solid line) and programming experience (dashed lines) classification accuracy compared against increasing bucket size. The data was split into even-sized buckets. Programming experience classification accuracies are shown for three different classifiers: Bayesian Network, Random Forest, and the majority class classifier ZeroR. The x-axis represents bucket size and the y-axis expresses identification and classification accuracy.

with identification, where the accuracy temporarily improved when transitioning from rounding to nearest 200 milliseconds to rounding to nearest 300 milliseconds. Overall, classification accuracy declines more slowly than identification accuracy with the rounding method. Similar to the rounding method, classification accuracies with the bucket method degrade with each step. A clear difference is that with the bucket method the classification accuracies decline faster, nearing the performance of the baseline ZeroR classifier when the bucket size is 600 ms. In contrast, with the rounding method, Bayesian Network and Random Forest outperform ZeroR by over 10 percentage points at the 600 millisecond mark. Nevertheless, classification accuracy does not decline as fast as identification accuracy with the bucket method – for example, with 300 millisecond buckets, reliable identification is no longer possible, but classification accuracy is still significantly better than with the majority class classifier. The decline in identification and classification accuracy is shown in Figure 1.

DISCUSSION AND CONCLUSIONS

In this work, we studied how typing profile data could be anonymized whilst retaining information important to researchers in the data. The motivation for the study is to be able to release open data sets where data that could be used to identify subjects is removed. We explored two different ways of anonymizing data consisting of student typing profiles on programming courses.

The results of our experiments indicate that it is possible to anonymize keystroke data in a way that preserves information relevant to researchers in our context. We showed that typing profiles based on keystroke data can still be used to classify programmers based on their programming experience, even when the data has been sufficiently anonymized so that programmers cannot be identified with reliable accuracy based on keystroke latencies.

For the rounding method, rounding keystroke average latencies to the nearest 500 milliseconds would be optimal in our context. When rounding to the nearest 500 milliseconds, reliable identification is hard. Approximately 7% of the students are correctly identified with a threshold of $k = 10$. This accuracy is very low when compared to the non-anonymized accuracy of around 98.5%. Purely random guessing would yield an identification accuracy of around 5% with our data set, which means that even with the 7% accuracy it is possible that there may still be some information on identity in the data, which might not be acceptable in all scenarios. With the same 500 millisecond rounding, programming experience can be inferred accurately for 73.9% students. With the Random Forest classifier, programming experience classification accuracy has remained the same as without anonymization, and with the Bayesian Network classifier, it declined only by 1.5 percentage points.

For the bucket method, the optimal amount of anonymization is quite different from the rounding method in our context. With even-sized 300 millisecond buckets, identification accuracy has decreased to the lowest value it will reach. At that point, programming experience classification is possible with around 71% accuracy compared to the 58.8% accuracy with the majority classifier. The result indicates that the bucket method is more efficient at anonymizing the data, although more domain-relevant information is lost in the process.

The results of the rounding method are interesting due to the fact that only keystroke latencies between 10 and 750 milliseconds were included in the typing profiles. When rounding to the nearest 500 milliseconds, there are only two possible values for the features – 0 milliseconds or 500 milliseconds – since all values between 0 and 250 milliseconds will be rounded to 0 milliseconds while values between 250 milliseconds and 750 milliseconds (the upper bound) will be rounded to 500 milliseconds. The result means that for inferring programming experience from typing profiles, it is sufficient to categorize all average latencies that the typing profiles include into two buckets based on whether the student is fast or slow at writing the digraph.

Another interesting find is that when the rounding method is used, identification seems quite reliable with an accuracy of around 74% even when rounding to the nearest 300 or 400 milliseconds. To further examine this, we plotted the changes in identification accuracies in 10 millisecond intervals. The resulting plot is in Figure 2. The local maxima for the two courses are at 340 ms with 86.3% accuracy and 360 ms with 90.2% accuracy. When rounding to both 340 and 360 milliseconds, there are only three buckets in our data due to filtering out events that are not between 10 and 750 ms. For exam-

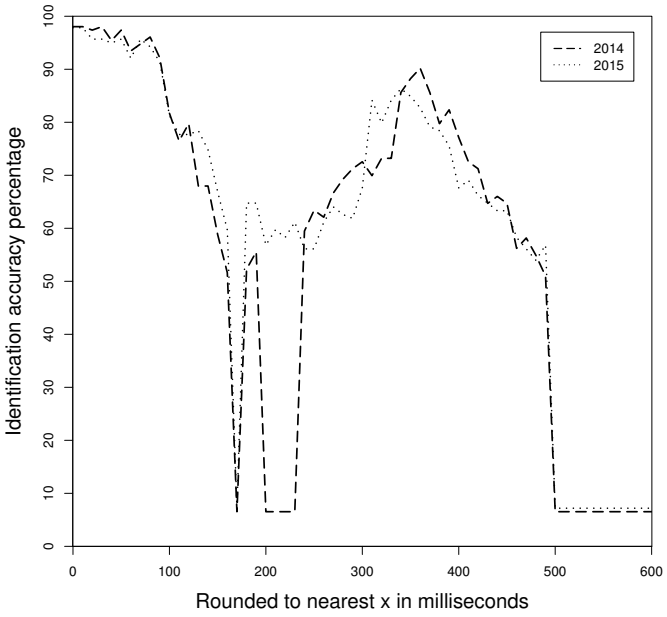

Figure 2. Identification accuracy compared against rounding precision. All values in the data were rounded to a nearest millisecond value. The larger the millisecond value in the x-axis, the lesser the rounding precision. The y-axis expresses identification accuracy.

ple, with 340 milliseconds, values between 0 and 170 ms are rounded to 0 ms, values between 170 and 510 ms are rounded to 340 ms, and values between 510 ms and 850 ms are rounded to 680 ms. This result suggests that fine-grained timestamp data is not actually necessary to identify programmers from their typing patterns. Only categorizing average keystroke latencies into three buckets – slow, mediocre, fast – might be enough for reliable identification.

The effect seen in Figure 2 implies that categorizing data into 3 buckets works better for identification than categorizing data into more buckets, unless the rounding starts to be insignificant (under 100 milliseconds). A potential explanation is that additional buckets beyond three add unnecessary noise to the data. For example, with five buckets – very slow, slow, mediocre, fast, very fast – there might not be enough average latencies in the very slow and very fast buckets. On the other hand, some average latencies that should be categorized to the mediocre bucket for maximal performance might be categorized to the slow or fast buckets.

Moreover, the observed effect is a cautionary result for researchers seeking to anonymize their data. Using a similar method and observing e.g. that the identification accuracies are low enough for sharing the data at the 200 millisecond point, and adding an additional 100 milliseconds "just to be sure", plenty of information that could be used to identify the individuals in the data would be shared accidentally.

The results of our studies show that keystroke timings can be anonymized in a way that retains informative value in data, and thus keystroke timings can be included in open data sets as long as proper anonymization procedures are followed. A limitation of our study is that there were only 199 and 153 students in our data sets. This is due to a language constraint as the courses were not organized in English. Future work should examine how the methodologies outlined in this work perform when in addition to large-scale data, the amount of students is larger. In addition, further research is needed to investigate whether other information than programming experience can be inferred from obscured data. Furthermore, future work should investigate how removing possible hidden identifiers other than keystroke latencies – such as text content – affect both identification accuracy and inference of valuable information.

ACKNOWLEDGMENTS

We are grateful to the anonymous reviewers whose comments helped us improve the paper. This work was partially funded by Academy of Finland under grant number 303694, *Skills, education and the future of work*.

REFERENCES

1. Ahmed Al-Zubidy, Jeffrey C Carver, Sarah Heckman, and Mark Sherriff. 2016. A (Updated) Review of Empiricism at the SIGCSE Technical Symposium. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*. ACM, 120–125.

2. Robert Bixler and Sidney D'Mello. 2013. Detecting Boredom and Engagement During Writing with Keystroke Analysis, Task Appraisals, and Stable Traits. In *Proc. of the 2013 International Conference on Intelligent User Interfaces (IUI '13)*. ACM, New York, NY, USA, 225–234. DOI: http://dx.doi.org/10.1145/2449396.2449426

3. Neil Christopher Charles Brown, Michael Kölling, Davin McCall, and Ian Utting. 2014. Blackbox: a large scale repository of novice programmers' activity. In *Proc. of the 45th ACM technical symposium on Computer science education*. ACM, 223–228.

4. Coursera. 2016. Coursera Signature Track. https://www.coursera.org/signature/. (2016). Accessed: 2016-10-24.

5. Jon P Daries, Justin Reich, Jim Waldo, Elise M Young, Jonathan Whittinghill, Andrew Dean Ho, Daniel Thomas Seaton, and Isaac Chuang. 2014. Privacy, anonymity, and big data in the social sciences. *Commun. ACM* 57, 9 (2014), 56–63.

6. Dataverse. 2016. The Dataverse Project. http://dataverse.org/. (2016). Accessed: 2016-10-24.

7. Paul S. Dowland and Steven M. Furnell. 2004. A Long-Term Trial of Keystroke Profiling Using Digraph, Trigraph and Keyword Latencies. In *Security and Protection in Information Processing Systems*, Yves Deswarte, Frédéric Cuppens, Sushil Jajodia, and Lingyu Wang (Eds.). IFIP - The International Federation for Information Processing, Vol. 147. Springer, 275–289. DOI:http://dx.doi.org/10.1007/1-4020-8143-X_18

8. Cynthia Dwork. 2008. Differential privacy: A survey of results. In *Theory and applications of models of computation*. Springer, 1–19.

9. Clayton Epp, Michael Lippold, and Regan L. Mandryk. 2011. Identifying Emotional States Using Keystroke Dynamics. In *Proc. of the SIGCHI Conference on Human Factors in Computing Systems (CHI '11)*. ACM, New York, NY, USA, 715–724. DOI: http://dx.doi.org/10.1145/1978942.1979046

10. Benjamin C. M. Fung, Ke Wang, Rui Chen, and Philip S. Yu. 2010. Privacy-preserving Data Publishing: A Survey of Recent Developments. *ACM Comput. Surv.* 42, 4, Article 14 (2010), 53 pages. DOI: http://dx.doi.org/10.1145/1749603.1749605

11. R Stockton Gaines, William Lisowski, S James Press, and Norman Shapiro. 1980. *Authentication by keystroke timing: Some preliminary results*. Technical Report.

12. Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. 2009. The WEKA data mining software: an update. *ACM SIGKDD explorations newsletter* 11, 1 (2009), 10–18.

13. Yeye He and Jeffrey F Naughton. 2009. Anonymization of set-valued data via top-down, local generalization. *Proc. of the VLDB Endowment* 2, 1 (2009), 934–945.

14. Petri Ihantola, Arto Vihavainen, Alireza Ahadi, Matthew Butler, Jürgen Börstler, Stephen H. Edwards, Essi Isohanni, Ari Korhonen, Andrew Petersen, Kelly Rivers, Miguel Ángel Rubio, Judy Sheard, Bronius Skupas, Jaime Spacco, Claudia Szabo, and Daniel Toll. 2015. Educational Data Mining and Learning Analytics in Programming: Literature Review and Case Studies. In *Proc. of the 2015 ITiCSE on Working Group Reports (ITICSE-WGR '15)*. ACM, New York, NY, USA, 41–63. DOI:http://dx.doi.org/10.1145/2858796.2858798

15. M. Karnan, M. Akila, and N. Krishnaraj. 2011. Biometric personal authentication using keystroke dynamics: A review. *Applied Soft Computing* 11, 2 (2011), 1565 – 1573. DOI: http://dx.doi.org/10.1016/j.asoc.2010.08.003 The Impact of Soft Computing for the Progress of Artificial Intelligence.

16. Georgios Kellaris and Stavros Papadopoulos. 2013. Practical differential privacy via grouping and smoothing. In *Proc. of the 39th international conference on Very Large Data Bases (PVLDB'13)*. VLDB Endowment, 301–312. http://dl.acm.org/citation.cfm?id=2488335.2488337

17. Kevin S. Killourhy and Roy A. Maxion. 2012. Free vs. Transcribed Text for Keystroke-dynamics Evaluations. In *Proc. of the 2012 Workshop on Learning from Authoritative Security Experiment Results (LASER '12)*. ACM, New York, NY, USA, 1–8. DOI: http://dx.doi.org/10.1145/2379616.2379617

18. Kenneth R Koedinger, Ryan SJd Baker, Kyle Cunningham, Alida Skogsholm, Brett Leber, and John Stamper. 2010. A data repository for the EDM community: The PSLC DataShop. *Handbook of educational data mining* 43 (2010).

19. Juho Leinonen, Krista Longi, Arto Klami, Alireza Ahadi, and Arto Vihavainen. 2016a. Typing Patterns and Authentication in Practical Programming Exams. In *Proc. of the 2016 ACM Conference on Innovation and Technology in Computer Science Education*. ACM, 160–165.

20. Juho Leinonen, Krista Longi, Arto Klami, and Arto Vihavainen. 2016b. Automatic Inference of Programming Performance and Experience from Typing Patterns. In *Proc. of the 47th ACM Technical Symposium on Computing Science Education (SIGCSE '16)*. ACM, New York, NY, USA, 132–137. DOI: http://dx.doi.org/10.1145/2839509.2844612

21. Krista Longi, Juho Leinonen, Henrik Nygren, Joni Salmi, Arto Klami, and Arto Vihavainen. 2015. Identification of Programmers from Typing Patterns. In *Proc. of the 15th Koli Calling Conference on Computing Education Research*. ACM, 60–67.

22. J.V. Monaco, J.C. Stewart, Sung-Hyuk Cha, and C.C. Tappert. 2013. Behavioral biometric verification of student identity in online course assessment and authentication of authors in literary works. In *Biometrics: Theory, Applications and Systems (BTAS), 2013 IEEE Sixth International Conference on*. 1–8. DOI: http://dx.doi.org/10.1109/BTAS.2013.6712743

23. John V Monaco and Charles C Tappert. 2016. Obfuscating Keystroke Time Intervals to Avoid Identification and Impersonation. In *The 9th IAPR International Conference on Biometrics (ICB)*. IEEE.

24. Fabian Monrose and Aviel Rubin. 1997. Authentication via Keystroke Dynamics. In *Proc. of the 4th ACM Conference on Computer and Communications Security (CCS '97)*. ACM, New York, NY, USA, 48–56. DOI: http://dx.doi.org/10.1145/266420.266434

25. Ruoming Pang, Mark Allman, Vern Paxson, and Jason Lee. 2006. The Devil and Packet Trace Anonymization. *SIGCOMM Comput. Commun. Rev.* 36, 1 (Jan. 2006), 29–38. DOI:http://dx.doi.org/10.1145/1111322.1111330

26. Pierangela Samarati and Latanya Sweeney. 1998. Generalizing Data to Provide Anonymity when Disclosing Information (Abstract). In *Proc. of the Seventeenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS '98)*. ACM, New York, NY, USA, 188–. DOI: http://dx.doi.org/10.1145/275487.275508

27. J.C. Stewart, J.V. Monaco, Sung-Hyuk Cha, and C.C. Tappert. 2011. An investigation of keystroke and stylometry traits for authenticating online test takers. In *Biometrics (IJCB), 2011 Int. Joint Conference on*. 1–7. DOI:http://dx.doi.org/10.1109/IJCB.2011.6117480

28. Yan Sun and Shambhu Upadhyaya. 2015. Secure and privacy preserving data processing support for active authentication. *Information Systems Frontiers* 17, 5 (2015), 1007–1015. DOI: http://dx.doi.org/10.1007/s10796-015-9587-9

29. Richard C Thomas, Amela Karahasanovic, and Gregor E Kennedy. 2005. An investigation into keystroke latency metrics as an indicator of programming performance. In *Proceedings of the 7th Australasian conference on Computing education-Volume 42*. Australian Computer Society, Inc., 127–134.

30. Arto Vihavainen, Juha Helminen, and Petri Ihantola. 2014a. How novices tackle their first lines of code in an IDE: analysis of programming session traces. In *Proc. of the 14th Koli Calling International Conference on Computing Education Research*. ACM, 109–116.

31. Arto Vihavainen, Matti Luukkainen, and Petri Ihantola. 2014b. Analysis of Source Code Snapshot Granularity Levels. In *Proc. of the 15th Annual Conference on Information Technology Education (SIGITE '14)*. ACM, New York, NY, USA, 21–26. DOI: http://dx.doi.org/10.1145/2656450.2656473

32. Arto Vihavainen, Thomas Vikberg, Matti Luukkainen, and Martin Pärtel. 2013. Scaffolding students' learning using Test My Code. In *Proc. of the 18th ACM conference on Innovation and technology in computer science education*. ACM, 117–122.

33. Lisa M. Vizer, Lina Zhou, and Andrew Sears. 2009. Automated Stress Detection Using Keystroke and Linguistic Features: An Exploratory Study. *Int. J. Hum.-Comput. Stud.* 67, 10 (Oct. 2009), 870–886. DOI: http://dx.doi.org/10.1016/j.ijhcs.2009.07.005